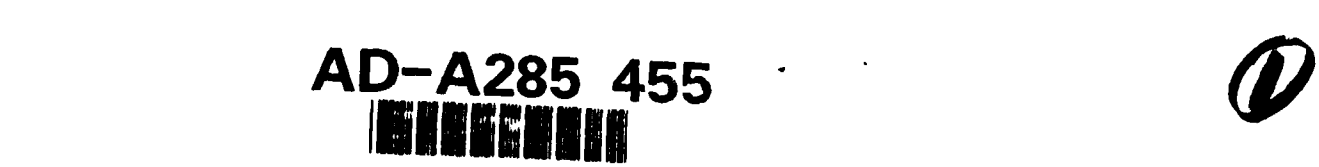


DSTO

Data Acquisition and Processing Software for the Low Speed Wind Tunnel Tests of the Jindivik Auxiliary Air Intake

S.S.W. Lam

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

Data Acquisition and Processing Software for the Low Speed Wind Tunnel Tests of the Jindivik Auxiliary Air Intake

S.S.W. Lam

Aeronautical and Maritime Research Laboratory

ABSTRACT

Technical Report

Data acquisition software has been developed for the wind tunnel tests of the auxiliary air intake of the Jindivik pilotless target aircraft in the AMRL Low Speed Wind Tunnel (LSWT) using the new Pressure Systems Incorporated (PSI) 8400 Measurement System under the control of an IBM PS/2 computer. The recent upgrade of the data acquisition system for the LSWT and the replacement of the mechanical Scanivalve pressure measuring system with PSI electronic pressure scanners has required major modifications to be made to existing software. To minimise the changes needed and to provide compatibility with data processing for previous tests the data acquired by the PSI electronic pressure equipment are transferred to the dedicated LSWT DEC PDP-11/44 mini-computer for storage and processing. This report describes the development and operation of new software for the LSWT tests of the Jindivik auxiliary air intake.

APPROVED FOR PUBLIC RELEASE

DSTO-TR-0043

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

Published by

DSTO Aeronautical and Maritime Research Laboratory
GPO Box 4331
Melbourne Victoria 3001 Australia

Telephone: (03) 626 7000
Fax: (03) 626 7999

AR No. 007-100
AUGUST 1994

APPROVED FOR PUBLIC RELEASE

Data Acquisition and Processing Software for the Low Speed Wind Tunnel Tests of the Jindivik Auxiliary Air Intake

EXECUTIVE SUMMARY

In May 1990 an experimental test program was conducted in the ARL Low Speed Wind Tunnel to evaluate various configurations of the auxiliary air intake in a Jindivik target aircraft model. The results showed that, with appropriate design of the auxiliary intake, significant improvement in the engine performance of the aircraft during take-off can be achieved. This led to a production design of the concept which had been modified to include three swivelling louvres that could be opened at take-off and closed in cruise flight. To verify the performance of the production design a second wind tunnel test program was conducted.

Since the first test program, the scanivalve mechanical pressure measuring equipment in the Low Speed Wind Tunnel has been replaced with a sophisticated and more efficient high speed electronic pressure measuring system which operates under the control of an IBM PS/2 personal computer. New data acquisition software needed to be developed for the second test program to:

1. collect and record the experimental data from the new pressure measuring equipment;
2. convert the data into a format which is compatible with the previous test results;
3. transfer the test data from the IBM PS/2 personal computer to the DEC PDP-11/44 mini-computer for processing and storage.

The new data acquisition system also required the existing software on the PDP-11/44 computer to be modified to process the new test data. The major modifications to the data processing software involved:

1. changes in the pressure calibration equations to convert the raw pressure data into the correct units;
2. the inclusion of logic statements to distinguish the previous test data from the new data so that the software could process both the old and the new test results;
3. changes in output routines that generate hard copy outputs from the new HP LaserJet III laser printer instead of the HP 7220T pen plotter.

In this report the functions of the new data acquisition software are described, and the operational procedures to be adopted when using the software are explained. Details of the modifications to the existing data processing software on the PDP-11/44 computer are also documented.

Author

S.S.W. Lam
Air Operations Division

Stephen S.W. Lam graduated in 1979 from the University of Melbourne with an honours Degree in Mechanical Engineering. He was admitted to the Degree of Master of Engineering Science at the same University in 1983 having undertaken experimental research and computer modelling on the dynamic response of the 'Salter Duck' ocean wave energy absorption device. In July 1983 he began a postgraduate research study on natural convection in trapezoidal cavity at Monash University and was awarded the Degree of Doctor of Philosophy in 1990. Since commencing employment at the then Aeronautical Research Laboratory in 1988, Stephen has been actively involved in the implementation of the new data acquisition systems in both the Low Speed Wind Tunnel and the Transonic Wind Tunnel. He is currently undertaking research in improving wind tunnel testing techniques and is involved in the implementation of the new strain gauge balance calibration facility.

CONTENTS

NOTATION

DC_{60}	Distortion factor across the engine face
$\dot{m}$	Engine air mass flow rate, kg/s
$\dot{m}\sqrt{T_o}/P_o$	Engine face air mass flow parameter
N	Engine rotational speed, RPM
P_1	Pressure before the venturi throat, kPa
P_a	Atmospheric pressure, kPa
P_o	Wind tunnel free stream total pressure, kPa
P_s	Static pressure, kPa
P_t	Total pressure at the engine face, kPa
$(P_t/P_o)_{avg}$	Pressure recovery (average of P_t/P_o at the engine face)
T_1	Temperature at the venturi, K
T_o	Wind tunnel free stream temperature, K
V	Nominal test airspeed, m/s
Vel	Actual test airspeed, m/s
x	Distance of static pressure probe from the engine face along the intake duct, m
α	Model pitch angle, deg.
β	Model yaw angle, deg.
ΔP	Pressure drop across the venturi throat, kPa
ϵ	Expansibility factor
λ	Venturi pressure ratio
ρ_v	Density of air at the venturi, kg/m^3

Subscripts

1	Conditions at the venturi meter
60	Average value over 60° sector at 5° intervals
avg	Average value over the engine face
a	Atmospheric value
o	Wind tunnel free stream total value
s	Static value
t	Total value at the engine face

1 INTRODUCTION

A research and development program to improve the take-off performance of the Jindivik target aircraft was carried out at ARL during 1990 and 1991 [1] to explore the concept of an auxiliary air intake system which would be deployed during take-off and closed at cruise. An experimental test program to evaluate various configurations of the auxiliary air intake was conducted in the 2.7 × 2.1 m Low Speed Wind Tunnel (LSWT) in May 1990 [2]. Significant improvements in the engine performance during take-off were achieved and this led to a production design of the concept. A second wind tunnel test program was carried out in November 1991 to confirm the performance of the production design which had been modified to include three swivelling louvres that can be opened at take-off and closed in flight.

In the first wind tunnel test program [2], the pressures at the engine face and along the air intake duct were measured using Scanivalves and the data were acquired and processed on a DEC PDP-11/44[1] minicomputer. Since then, a new data acquisition system has been installed in the LSWT and the Scanivalves have been replaced by a Pressure Systems Incorporated (PSI) 8400 Measurement System under the control of an IBM PS/2[2] personal computer via a National Instruments MC-GPIB adaptor. These upgrades required major changes to be made to the data acquisition and processing software for the second test program even though the test set-up was basically the same as the first one.

To minimise changes to the existing data processing software which was described in Reference 4, the data acquired by the IBM PS/2 were transferred to the PDP-11 computer via an RS-232 serial line and stored in the same format as in the previous test. Modifications to the software were made in such a way that previous test data may also be processed by the new versions of the data processing programs.

This report describes the development and operation of the new software for testing the Jindivik auxiliary air intake in the LSWT.

2 EXPERIMENTAL SET-UP

A 1/4 scale model of the Jindivik air intake system was mounted in the centre of the LSWT test section as shown in Figure 1. The engine air flow was simulated with a remote duplex blower which sucked air through the model inlet duct via flexible steel coil reinforced tubing and a tilted steel down pipe which passed through the floor of the test section to a venturi flow meter.

Total pressure measurements at the engine face were made with a 30 probe rake [2]. The orientation of the probes and their numeral designations are shown in Figure 2. Internal static pressures were measured with tubes tapped into the duct wall at right angles and fitted flush with the surface at 15 axial locations along the air

[1] DEC and PDP-11 are registered trade marks of Digital Equipment Corporation

[2] IBM and PS/2 are registered trademarks of International Business Machines Corporation

intake duct, and 6 tapping points equally spaced around the engine face location in the duct, as shown in Figures 2 and 3. These tappings were connected to two 32-port pressure scanners of the PSI 8400 Measurement System which was used to collect the pressure data.

3 DATA ACQUISITION SYSTEM

The data acquisition system was based on a PSI 8400 Measurement System under the control of an IBM PS/2 computer via a National Instruments GPIB interface adaptor. Figure 4 shows a schematic layout of the data acquisition system. The serial port of the PS/2 was connected to terminal port TT5 of a DEC PDP-11/44 mini-computer to allow data to be transferred to the PDP-11 for storage and processing. A dumb terminal connected to terminal port TT6 of the PDP-11 was used to input the air flow parameters at the venturi meter (in the previous test program, these parameters were read out by an operator through a microphone, over a high background noise level, to another operator at the wind tunnel control room). The hardcopy outputs of the test results were printed from a Hewlett-Packard (HP) LaserJet III laser printer connected to terminal port TT7 of the PDP-11. The 8-pages-per-minute laser printer, which replaced the sluggish HP 7220T pen plotter that was used in the previous test program, significantly reduced the printing time of the test results.

For proper communications between the PDP-11 and the PS/2, the serial line connecting the two computers had to be set at a baud rate of 4800. On the PDP-11 side, this was set with the command:

```
SET TERMINAL/SPEED=4800/NOECHO TT5:
```

At the PS/2 end, the line speed was set by the program JIN1 (see section 4.1). The terminal lines TT6 and TT7 of the PDP-11 were set at 9600 baud with the commands:

```
SET TERMINAL/SPEED=9600/NOSLAVE/ECHO TT6:
SET TERMINAL/SPEED=9600/NOSLAVE/ECHO TT7:
```

to match the communication speeds of the dumb terminal at the Venturi meter and the HP LaserJet III laser printer respectively. A detailed description of the experimental set-up is given in Reference 3.

4 DATA ACQUISITION SOFTWARE

The basic data acquisition and control program for the 8400 Measurement System on the IBM PS/2 is the Macro Command Processor (MCP). The configurations of the measurement system — such as the number of pressure ports, the calibration specifications for the pressure scanners, the manner in which the pressure data are

to be collected and so on — are coded in a *macro file*. Each macro consists of a sequence of statements which are either PSI System 8400 commands (see Chapter 5 of the *PSI System 8400 Users Manual*) or instructions to the program MCP to perform specific tasks. The macros used in the Jindivik auxiliary air intake test are contained in the file JINDIVIK.MAC, which is listed in full in Appendix A. Explanations of the commands are given as comments (lines beginning with the single quote character) throughout the file. Two of the more important macros defined in JINDIVIK.MAC are considered in some detail in the following:

1. **Init** — The initialisation macro. This macro sets up pressure port configurations and the calibration specifications for the transducers. Important items incorporated into the macro were that:
   - two 5-psid pressure scanners were employed,
   - all 32 ports of scanner 1 and the first 27 ports of scanner 2 were used,
   - the pressure transducers were to be calibrated at -5.0, -3.0, -1.0, 0.0, 1.0 psid,
   - when acquiring data, each pressure value was to be averaged over 50 readings taken at 5 millisecond intervals.
2. **Data** — The get-data macro. At the start of the macro, the user was prompted to enter a data file name and a line of description for the test point. It then executed the program JIN1 to initialise communications with the PDP-11 and started the program JINDIVIK on the PDP-11. The program JINDIVIK would prompt the operator at the venturi to input the flow parameters of the blower before returning program execution control to MCP on the PS/2, and then the command to acquire data was issued to the 8400 Measurement System. After receiving the averaged pressure data value from the 8400 System, the macro executed another program JIN2 to pass the results to the program JINDIVIK running on the PDP-11 and to write the results to data files.

4.1 Programs JIN1 and JIN2

The programs JIN1 and JIN2, written in Turbo C[3] programming language on the PS/2, were responsible for exchanging information and transferring data to and from the PDP-11. At the start of the **Data** macro, after the user had entered the data file name and test description, JIN1 would be executed to establish the link between the PS/2 and the PDP-11. The major functions of JIN1 were to:

1. Initialise the serial port (COM1:) of the PS/2 to 4800 baud and set up the necessary parameters and routines for successful communications over the serial line. The source codes of the communication routines were contained in a separate file JINCOMM.C.

[3]Turbo C is a registered trade mark of Borland International Inc.

2. Start the program **JINDIVIK** on the PDP-11 by sending the following PDP-11 commands down the serial line:

```
set term/noecho
run jindivik
```

3. Obtain the following test parameters from the operator in the wind tunnel control room:
   - the attitude (α and β) of the model,
   - atmospheric pressure, and
   - tunnel air temperature.
4. Send the name of the data file and the day's date to the PDP-11.
5. Wait for the **JINDIVIK** program on the PDP-11 to return with air flow parameters of the blower before exiting and returning control to **MCP**.

After the **MCP** regained control of the program execution from **JIN1**, it sent the *acquire-data* command to the 8400 Measurement System to obtain the pressure values (in lbf/in^2) from the scanners. It then executed the program **JIN2** in which each pressure value was multiplied by 1000 and converted to a 4-digit integer before being sent to the PDP-11 to be stored on an RL02 disk The conversion of the pressure data into integer format was to made comply with the data format of the previous test program [4].

The program **JIN1** was built from the code in the source files **JIN1.C**, **JINCOMM.C** and **SERIAL.H**. The source code for **JIN2** is contained in **JIN2.C**, **JINCOMM.C** and **SERIAL.H**. These files are listed in Appendices B through to E for reference.

4.2 Program JINDIVIK

The program **JINDIVIK** was written in FORTRAN on the PDP-11. It was executed by the program **JIN1** from the PS/2 via the serial communication line at the beginning of each *acquire-data* command. After receiving the data file name and the day's date from the PS/2 sent to it by **JIN1**, the **JINDIVIK** program prompted the operator at the venturi meter to enter the air flow parameters:

- upstream pressure, P_1,
- upstream temperature, T_1, and
- pressure drop across the venturi, ΔP,

via the keyboard of the dumb terminal connected to port **TT6** of the PDP-11 computer. It then signalled to the program **JIN1** on the PS/2 that all parameters at the Venturi had been entered and that acquisition of the pressure data could proceed,

and then waited until pressure data arrived from the PS/2. The relevant data of the test were written to two separate files (with extensions .DAT and .JCS) of different formats [4]. The .DAT file was in the the conventional LSWT pressure measurement data file format and was not used in the present analysis. The format of the .JCS file was the same as that described in Reference 4 and was used as an input file by programs CALJIN, PLTPS, PLTPT and PLTCNT [4] for processing. Both data files were written to the directory DL1:[100,7] on an RL02 disk with the label JINAIR. The source code for the program JINDIVIK is contained in the file JINDIVIK.FTN and is listed in Appendix F.

5 MODIFICATIONS TO THE DATA PROCESSING SOFTWARE

As in the previous test [4], the data at each test point were processed by the program CALJIN. The programs PLTPS, PLTPT and PLTCNT [4] were used to produce hardcopy graphical outputs in the form of:

- a graph of the ratio of intake static pressure (P_s) to tunnel freestream total pressure (P_o) versus the axial location in metres (x) along the air intake duct away from the engine face towards the intake lip. An example of this type of graphical output is given in Figure 5,
- a graph of the average ratio of total pressure at the engine face (P_t) to freestream total pressure (P_o) versus the normalized mass flow rate ($\dot{m}\sqrt{T_o}/P_o$), as shown in Figure 6, and
- a cross-sectional contour plot of P_t/P_o at the engine face, as shown in Figure 7.

The definition and method of calculating the distortion factor DC_{60} appearing in the outputs are described in detail in Reference 4.

Although every endeavour was made to maintain data and software compatibility with the previous series of tests, the hardware configuration of the new data acquisition system necessitated modifications to the above programs. There were two major areas of modifications; viz, processing of raw pressure data and printer output routines.

5.1 Processing of Raw Pressure Data

The main program for processing the data of the Jindivik auxiliary air intake test is CALJIN. The raw transducer values (averaged over 50 readings) were input from the data file with the name in the form "JINxxx.JCS" where xxx is the test point number, as in the first series of tests.

In the previous Scanivalve system, a set of four calibration equations were required to convert the transducer readings into real pressure values. Pressure ports 3 and 4

were connected to a constant pressure source of 21.0 kPa, equivalent to the scanivalve transducer reading of 590 approximately. This was assigned to the variables JPRESS(3) and JPRESS(4) in the program and used in the calibration calculations (see Section 2.2.2 of Reference 4). The 8400 Measurement System performed the calibration internally and the pressure values were converted into lbf/in^2 before being sent to the PS/2 where they were multiplied by 1000 and stored in the data file. Hence when processing the data obtained with the new system, no calibration equation was needed except the conversion of the pressure values from $\times 1000$ lbf/in^2 to Pa. Pressure ports 3 and 4 were connected, in this case, to a constant pressure source in the range of 200 to 300 Pa (equivalent to 0.029 to 0.044 lbf/in^2, and recorded as an integer value in the range of 29 to 44 in the data files). The comparatively small values from these 2 pressure ports were used as a convenient indication that the 8400 Measurement System was in use and that the new conversion scheme is to be employed. The old Scanivalve system conversion scheme is retained in the program so that data from previous tests can also be processed.

There are two major modifications to the calculations of the engine mass flow parameter, $\dot{m}\sqrt{T_o}/P_o$. Firstly, the mass flow rate, $\dot{m}$, which was expressed in lb/s in the previous test, is converted to kg/s in the new software. Secondly, the old software used atmospheric temperature and pressure in the expression $\dot{m}\sqrt{T_o}/P_o$. In the present test, the total temperature and pressure in the test section of the wind tunnel are used instead. The details of the equations used in the calculation of the air mass flow rate are given in Appendix G.

In the graph of $(P_t/P_o)_{avg}$ versus $\dot{m}\sqrt{T_o}/P_o$, two straight lines representing constant engine speeds of 12000 and 13800 RPM were represented respectively by the equations:

$$\frac{\dot{m}\sqrt{T_o}}{P_o} = 0.1780\left(\frac{P_t}{P_{o\,avg}}\right)$$
$$\frac{\dot{m}\sqrt{T_o}}{P_o} = 0.2093\left(\frac{P_t}{P_{o\,avg}}\right).$$

These two equations are different from those given in page 13 of Reference 4 because the units of $\dot{m}$ and pressure in the previous tests are in lb/s and lbf/in^2 respectively, but in the new tests the units of $\dot{m}$ and pressure are in kg/s and kPa.

5.2 Printer Output Routines

After processing each test data point the program CALJIN produced a one-page summary of the results (an example of which is shown in Figure 8, the definitions of the symbols appeared in the printout are explained in Reference 4) on a LaserJet III laser printer. This was made possible by attaching a string of PCL[4] context printer commands [5] at the beginning of the print file, which set up the character font specifications and page margins. These commands were:

[4]PCL is a registered trade mark of Hewlett-Packard Company.

<ESC>E — reset printer,
<ESC>(8U — select Roman-8 font style,
<ESC>(s0P — fixed character spacing,
<ESC>(s12H — 12 character per inch,
<ESC>(s10V — 10 points character size,
<ESC>(s0S — upright characters,
<ESC>(s0B — normal character stroke,
<ESC>(s3T — courier typeface,
<ESC>\&a10L — left margin = 10,
<ESC>\&a5M — right margin = 5,
<ESC>\&l4E — top margin = 4,
<ESC>\&l60F — 60 lines per page,
<ESC>\&l0O — portrait orientation.

where <ESC> is the "escape" character code (ASCII value 27). These commands may be put into a condensed form by grouping and combining them, as described in the *LaserJet III Technical Reference Manual* [5], in the following manner:

```
<ESC>E
<ESC>(8U
<ESC>(s0p12h10v0s0b3T
<ESC>&a10l5M
<ESC>&l4e60f0O
```

In the program, these commands were coded into the variable lprhed defined as an array of BYTE. The ASCII values of the command characters were assigned into the array using a DATA statement, and were written to the output print file as character variables:

```
      BYTE lprhed(45)
      DATA lprhed/27,69,27,40,56,85,27,40,115,48,112,49,50,104,49,
     1     48,118,48,115,48,98,48,51,84,27,38,97,49,48,108,53,77,
     2     27,38,108,52,69,27,38,108,54,48,102,48,79/
      .
      .
      .
      WRITE ( 1,400 ) lprhed
400   FORMAT ( 45A1 )
```

To ensure that the printed page would be ejected from the printer the command string <FF><ESC>E, where <FF> is the form-feed character (ASCII value 12), was written to the print file at the end of the file. This command string was coded into the variable array lprend as follows:

```
      BYTE lprend(3)
      DATA lprend/12,27,69/
```

The programs PLTPS, PLTPT, and PLTCNT used the Hewlett-Packard Graphic Language (HP-GL) instruction set to generate hardcopy graphical outputs on an HP 7220T pen plotter. The HP LaserJet III laser printer accepted the HP-GL/2 instruction set which is a new version of HP-GL. While there were differences between the two versions, the graphic instructions generated by PLTPS, PLTPT, and PLTCNT were compatible with HP-GL/2. The HP plotting library that was used to generate these instructions, however, issued certain machine dependent codes during the initialisation and closing routines, which were directed specifically to HP pen plotters and which were not recognised by the LaserJet printer. The two routines, HPINIT and PLOT of the HP plotting library, were therefore modified to eliminate the output statements which generated these machine dependent codes. The HP LaserJet III printer was initialised instead with command strings in a format similar to that used in CALJIM.

The initialisation command strings used in PLTPS, PLTPT, and PLTCNT were slightly different because of the difference in sizes and orientations of the plots they generated. The commands used in PLTPS were:

<ESC>E	—	reset printer,
<ESC>\&l0O	—	portrait orientation,
<ESC>*p150x-900Y	—	position cursor at 150 dots[5] right and 900 dots down,
<ESC>*c0T	—	set picture frame anchor point,
<ESC>*c5400x7920Y	—	set picture frame width and height to be 5400 and 7920 decipoints[5] respectively,
<ESC>*c7.5k11L	—	set HP-GL/2 plot size to be 7.5 × 11 inches.
<ESC>\%1B	—	enter into HP-GL/2 graphic mode.

In PLTPT these commands were similar, as shown in the following:

<ESC>E	—	reset printer,
<ESC>\&l0O	—	portrait orientation,
<ESC>*p60x-900Y	—	position cursor at 60 dots right and 900 dots down,
<ESC>*c0T	—	set picture frame anchor point,
<ESC>*c5400x7220Y	—	set picture frame size to 5400 × 7220 decipoints,
<ESC>*c7.5k11L	—	set HP-GL/2 plot size to 7.5 × 11 inches.
<ESC>\%1B	—	enter into HP-GL/2 graphic mode.

In PLTCNT these commands were identical to those in PLTPS except without the cursor positioning and set anchor point instructions:

<ESC>E	—	reset printer,
<ESC>\&l0O	—	portrait orientation,
<ESC>*c5400x7920Y	—	set picture frame size to 5400 × 7920 decipoints,
<ESC>*c7.5k11L	—	set HP-GL/2 plot size to be 7.5 × 11 inches.
<ESC>\%1B	—	enter into HP-GL/2 graphic mode.

[5] In the HP LaserJet III printer one dot equals $\frac{1}{300}$ inch and a decipoint is $\frac{1}{720}$ inch.

The following graphic commands were also incorporated into the three plotting programs to enhance the quality of the graphical outputs:

- Select a standard character set — this command was used to select the PC-8 symbol set (see Appendix A of Reference 5) with Univers typeface. The HP-GL/2 command for this selection is:

  ```
  SD1,341,2,1,4,17,7,52;
  ```

- Select an alternative character set — this was to allow the mathematical symbols (α, β and $\surd$) to be used. The command for this is:

  ```
  AD1,269,2,1,4,19,7,52;
  ```

- Define pen width — this command enabled different line thicknesses to be used for different types of line. This was particularly useful in distinguishing the contour lines from the engine surface boundaries in the contour plots. The pen width command format is:

  ```
  PWx
  ```

  where x is the width of the line in mm.

6 SOFTWARE INSTALLATION

The source files for the programs JINDIVIK, CALJIN, PLTPS, PLTCNT and PLTPT are stored on a floppy diskette which is kept in the Applied Aerodynamics Archive Area at ARL. The diskette is labelled with the description: "*JINDIVIK AUXILIARY AIR INTAKE - SERIES 2*".

The procedures to install these programs in the PDP-11 computer (assumming that the user has logged into the PDP-11 with the appropriate account details) are:

1. Change the default directory to [200,10] with the following command:

   ```
   SET DEFAULT [200,10]
   ```

2. Place the floppy diskette containing the source files into the floppy disk drive 0 of the PDP-11 computer. Enter the following commands to copy the files from the diskette to the current directory:

   ```
   MOUNT/OVERRIDE DY0:
   COPY DY0:[200,10]*.* *
   ```

3. After all the files have been copied, execute the command:

```
@INSTALL
```

to start the installation procedure.

The source files for the programs `JIN1` and `JIN2` are stored on an IBM format 3.5-inch floppy diskette labelled with the description "*JINDIVIK AUXILIARY AIR INTAKE - SERIES 2*". The diskette is also kept in the Applied Aerodynamics Archive Area. To install the programs, place the diskette in drive A of the PS/2 computer and enter the command:

```
A:\INSTALL
```

The programs `JIN1` and `JIN2` will then be located in the directory `D:\LSWT\JINDIVIK` of the PS/2.

7 CONCLUSIONS

The upgrade of the data acquisition system for the ARL Low Speed Wind Tunnel has required major modifications to be made to the software for the acquisition and processing of data for the later series of Jindivik auxiliary air intake tests. To maintain software compatibility with previous tests, data were stored on the PDP-11 minicomputer in exactly the same way as before and were processed with the same set of programs modified to take into account the new hardware configuration.

The replacement of the HP 7220T pen plotter with an HP LaserJet III laser printer has significantly reduced the time to produce the hardcopy outputs of the test results and enhanced the quality of these outputs.

The software was modified in such a way that data from the previous test series can still be processed by the new versions of the programs.

REFERENCES

[1] Abdel-Fattah, A. M. *Modification of Jindivik air intake duct with an auxiliary intake – static aerodynamic tests.* Propulsion Technical Memorandum 460, Aeronautical Research Laboratory, DSTO Australia, August 1991.

[2] Abdel-Fattah, A. M. and Link, Y. Y. *Low Speed Wind Tunnel tests on Jindivik air intake duct with and without an auxiliary intake.* Propulsion Technical Memorandum 472, Aeronautical Research Laboratory, DSTO Australia, March 1992.

[3] Lam, S. S. W. *Wind Tunnel Tests of an auxiliary intake system with swivelling louvres fitted to Jindivik target aircraft.* ARL Technical Note, to be published.

[4] Link, Y. Y. *A FORTRAN program for processing Low Speed Wind Tunnel test data for the Jindivik auxiliary intake.* Flight Mechanics Technical Memorandum 426, Aeronautical Research Laboratory, DSTO Australia, September 1990.

[5] *LaserJet III Technical Reference Manual.* HP Part No. 33449 - 90903, Hewlett-Packard Company, US, March 1990.

APPENDIX A

THE MACRO FILE JINDIVIK.MAC

```
title JINDIVIK TEST MAIN MENU
'
' Stephen S.W. Lam
' 07-OCT-1991
'
macro Help H
clrscr
print ********************************************
print * Init: Initializes the System Unit 8400   *
print * Cal : Calibrate Transducers               *
print * Data: Get Data                            *
print * Plot: Plot Data                           *
print ********************************************
endm

macro Init I
print Initializing System 8400 ...
'
' Disable command echo on front panel
'
SC4 0
'
' Scanner Digitizer Unit: 2 scanners of 32 ports each
'
sd1 111 1-2 32 1
'
'-------------------------------------------------------------
' Data acquisition parameters
'   111 : Address of the SDU
'     1 : Table number -- defines a particular test set up
'    50 : Number of frames to be averaged
'  5000 : Delay between frame readings (in microseconds)
'     1 : Number of measurement sets
'     0 : Delay between measurements sets (in milliseconds)
'  free : Trigger mode -- Software trigger
'   seq : Scan address mode -- sequential
'     2 : Data output format -- 4 byte, Engineering Units
'-------------------------------------------------------------
sd2 111 1 50 5000 1 0 free seq 2
'
' Define another scan table (2) to output data continuously
'
sd2 111 2 1 0 1 0 free seq 2
'-------------------------------------------------------------------------
' Setup the number of ports to be scanned on each scanner
'     111 : Address of the SDU
'       1 : Scan table number -- defines the list of ports for scanning
' 101-132 : Scanner 1, port 1 to port 32
' 201-222 : Scanner 2, port 1 to port 27
'-------------------------------------------------------------------------
sd3 111 1 101-132 201-227
sd3 111 2 101-132 201-227
'
'-----------------------------------------------
' Setup the Pressure Calibration Unit
'   211 : Address of the PCU (Cluster 2, Rack 1, Slot 1)
```

```
'    1  : Logical range number 1
'  DIFF : Differential pressure mode
' 0.0001 : Tolerance of calibration is 0.0001 psia
'   20  : Maximum Pressure of the unit = 20 PSI absolute
'-------------------------------------------------
pc1 211 1 DIFF 0.0001 20
printbuffer
'
'-------------------------------------------------
' Setup calibration pressure values
' 211 : Address of the PCU (Cluster 2, Rack 1, Slot 1)
' 5 calibration points: -5.0 to 1.0 psid
'-------------------------------------------------
pc2 211 -5.0 -3.0 -1.0 0.0 1.0
'-------------------------------------------------------------
' Select the pressure values of the PCU to be in psi unit (1)
'-------------------------------------------------------------
pc4 211 1
'-------------------------------------------------
' Set Pulse duration to be 5 seconds for shifting
' the calibration valve
'-------------------------------------------------
cp1 5
'-------------------------------------------------
' Set 10 seconds delay time to wait for calibration
' pressures to stabilize during calibration
'-------------------------------------------------
cp2 10
endm

macro Calib C
print
print Performing Full Calibration
print Please wait ...
ca3 1
endm

macro Data D
' ---------------------------------------------------------------------
'  Get-Data Command:
'     1. Obtain data file name and a line of description
'     2. Spawn JIN1 to set up communication with PDP-11
'     3. Get data from the 8400 Measurement System
'     4. Spawn JIN2 to send data to PDP-11 for storage and processing
'---------------------------------------------------------------------
input Enter data file name:
input Enter test description:
output fopen data\%1
output fprint ## %2
date
output system jin1 %1 "%2"
ad2 1
fclose
output system jin2 data\%1
endm
```

APPENDIX B

THE PROGRAM SOURCE FILE JIN1.C

```
/* ---------------------------- FILE: JIN1.C ------------------------------ **/
/** Program for the data acquisition of Jindivik Auxiliary Air Intake Wind **/
/** Tunnel Test - Series 2 **/
/** Executed by MCP, run in conjunction with JINDIVIK in PDP-11/44 **/
/** **/
/** Written by Stephen S.W. Lam **/
/* ------------------------------------------------------------------------ **/
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <dos.h>

#define MAX_LINE 127

extern char *nextin, *nextout; /* circular buffer for serial communications */
char data_file_name[40],
     desc[MAX_LINE];          /* test description */

void get_parameter( void );
void send_test_info( void );
int get_line( char *s, int lim);

void main( int narg, char *argv[] )
{
/* ------------------------------------------------------------ --------- */
/* data file name is given as the first argument in the command line and */
/* test description (desc) is the second.                                */
/* --------------------------------------------------------------------- */
    strcpy ( data_file_name, *++argv );
    strcpy ( desc, *++argv );

/* --------------------------------------------------------------------- */
/*  Initialise COM port and send PDP-11 commands to start up JINDIVIK    */
/* --------------------------------------------------------------------- */
    initcomport();
    send_line("set term/noecho");
    send_line("r jindivik");
/* --------------------------------------------------------------------- */
/*  Get test parameters from the operator at the Control Room and receive*/
/*  air flow parameters entered by the operator at the Venturi via PDP-11*/
/* --------------------------------------------------------------------- */
    get_parameter();
/* --------------------------------------------------------------------- */
/*  Restore serial interrupt and return program control back to MCP      */
/* --------------------------------------------------------------------- */
    restore_serialint();
}

void get_parameter ( void )
{
    int c, i=0;
    char line[MAX_LINE];
    float jin_data[7];

/* --------------------------------------------------------------------- */
/*  Get parameters from operator                                         */
```

```
/* ------------------------------------------------------------------ */
    printf(" Alpha (Deg.) = ");
    get_line(line, MAX_LINE);
    sscanf(line, "%f", &jim_data[0]);

    printf(" Beta (Deg.) = ");
    get_line(line, MAX_LINE);
    sscanf(line, "%f", &jim_data[1]);

    printf(" Atmospheric Pressure (kPa) = ");
    get_line(line, MAX_LINE);
    sscanf(line, "%f", &jim_data[2]);

    printf(" Tunnel Temperature (Deg. C) = ");
    get_line(line, MAX_LINE);
    sscanf(line, "%f", &jim_data[3]);
    jim_data[3] += 273.15;

/* ------------------------------------------------------------------ */
/*  Send the test descriptions and the above test parameters to PDP-11  */
/* ------------------------------------------------------------------ */
    printf(" Inputs of test parameter in progress, please wait ... ");
    send_line( desc );
    send_test_info();
/* ------------------------------------------------------------------ */
/*  Loop until flow parameters from Venturi meter come in via the      */
/*  the serial COM1 port                                               */
/* ------------------------------------------------------------------ */
    clear_serial_queue();
    do {
        if ( nextin != nextout )
        {
            c = readcomm();
            line[i++] = c;
        }
    } while ( c != 'C' );       /** End of data line signal **/

    line[i] = '\0';
    for ( i=0; i<=2; i++ )
        sscanf(line+i*7, "%f", &jim_data[i+4]);
    printf(" Venturi Readings:\n    T1 = %5.2f deg. C, P1 = %6.2f kPa\r\n",
           jim_data[4], jim_data[5]);
    printf("    Pressure Drop = %5.2f inches of water\n\n", jim_data[6]);
/* ------------------------------------------------------------------ */
/*  Tell PDP-11 to continue and send down data back to PDP-11 for storage*/
/* ------------------------------------------------------------------ */
    send_line ("C");
    for ( i=0; i<=3; i++)
        sprintf(line+i*7, " %6.2f", jim_data[i]);
    line[28]='\0';
    send_line( line );

    printf("Ready to scan pressure");
}

/* ------------------------------------------------------------------ */
/*  send_test_info: send test information to PDP-11                   */
/* ------------------------------------------------------------------ */
void send_test_info(void)
{
    char *months[] = {"JAN", "FEB", "MAR", "APR", "MAY", "JUN", "JUL",
                      "AUG", "SEP", "OCT", "NOV", "DEC"};
```

```
    char filnam[20], fildat[12], *fn_ptr;
    int len, i;
    struct date date;

    len = strlen(data_file_name) - strlen( strchr(data_file_name, '.') );
    for ( i=0; i <=len; i++ )
        *(filnam+i) = toupper(*(data_file_name+i));
    filnam[len] = '\0';
    send_line( filnam );

    getdate(&date);
    sprintf(fildat, "%d-%s-%d", date.da_day, months[date.da_mon-1],
           date.da_year);
    send_line( fildat );

}

/* ---------------------------------------------------------------------- */
/* get_line: get a line of characters from the console                    */
/* ---------------------------------------------------------------------- */
int get_line( char *s, int lim)
{
   char *s2;

   *s = lim;
   s2 = cgets( s);
   strcpy( s, s2);
   cprintf( "\r\n");
   return( strlen( s));
}
```

APPENDIX C

THE PROGRAM SOURCE FILE JIN2.C

```
/** ---------------------- FILE: JIN2.C ------------------------------- **/
/**  Program for the data acquisition of Jindivik Auxiliary Air Intake Wind **/
/**  Tunnel Test - Series 2                                              **/
/**  Executed by MCP, run in conjunction with JINDIVIK in PDP-11/44      **/
/**                                                                      **/
/**  Written by Stephen S.W. Lam                                         **/
/** ---------------------------------------------------------------------- **/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <dos.h>

int fget_line( char *s, int lim, FILE *fp);
void read_data_file( char *filename );

#define MAXPORT 64      /*** Maximum no. of pressure ports ***/
#define HEADINGS 4      /*** No. of lines of Headings ***/
#define DATA_WIDTH 9    /*** Maximum field width of Data item ***/
#define MAX_LINE 127    /*** Maximum length of line ***/

int ipress[MAXPORT];     /*** integer value of pressure data ***/
int nports=59;           /*** number of pressure ports ***/
extern char *nextin, *nextout;

void main( int narg, char *argv[] )
{
    char line1[MAX_LINE],line2[MAX_LINE],line3[MAX_LINE];
    int i, c;

    printf("\nSending test data to PDP-11 ...");
/* ---------------------------------------------------------------------- */
/* Initialise COM1 serial port.  Argument 1 on command line is the data  */
/* file name.  Use that as input argument to read_data_file fuction which*/
/* reads the pressure data acquired by MCP                                */
/* ---------------------------------------------------------------------- */
    initcomport();
    read_data_file( *++argv );

/* ---------------------------------------------------------------------- */
/* Put the pressure data on 3 separate lines before sending down to the  */
/* PDP-11 since the FORTRAN program JINDIVIK can read a maximum number   */
/* of 132 characters only per line.                                       */
/* ---------------------------------------------------------------------- */
    for ( i = 0; i <=19; i++ )
    {
        sprintf(line1+6*i, " %5d", ipress[i]);
        sprintf(line2+6*i, " %5d", ipress[i+20]);
        sprintf(line3+6*i, " %5d", ipress[i+40]);
    }
    send_line(line1);
    send_line(line2);
    send_line(line3);
/* ---------------------------------------------------------------------- */
/*  Wait until acknowleged by JINDIVIK on PDP-11                          */
/* ---------------------------------------------------------------------- */
```

```
    do
    {
        if ( nextin != nextout )
            c = readcomm();
    } while ( c != '>' );

/* ------------------------------------------------------------------- */
/*  Reset terminal characteristic and serial interrupt before exiting  */
/* ------------------------------------------------------------------- */
    send_line("set term/echo");
    restore_serialint();
    printf("\nDone\n\n");
}


/* ------------------------------------------------------------------- */
/* Read pressure data from file, pre-multiply them by 1000, and convert */
/* them to integers                                                    */
/* ------------------------------------------------------------------- */
void read_data_file( char *filename )
{
    FILE *datafp;
    int i, line_length, port_id;
    float psi_data;
    char line[MAX_LINE], c;

    datafp = fopen( filename, "r");

/* ------------------------------------------------------------------- */
/* Read Headings                                                       */
/* ------------------------------------------------------------------- */
    for (i=1; i<=HEADINGS; i++)
        fget_line( line, MAX_LINE, datafp );

    for(i=0; i <=nports-1; i++)
    {
       fscanf(datafp,"%d %f",&port_id, &psi_data);
       ipress[i] = psi_data*1000;
    }
    fclose(datafp);

}

/* ------------------------------------------------------------------- */
/* fget_line: get a line of characters from file fp                    */
/* ------------------------------------------------------------------- */
int fget_line( char *s, int lim, FILE *fp)
{
   if ( fgets( s, lim, fp) == NULL)
      return ( EOF);
   else {
      s[ strlen( s) - 1] = '\0';
      return( strlen( s));
   }
}
```

APPENDIX D

THE PROGRAM SOURCE FILE JINCOMM.C

```
/** ---------------------- FILE: JINCOMM.C ------------------------------ **/
/**  Serial communications functions called by JIN1 and JIN2              **/
/**                                                                       **/
/**  Modified from TCOMM.C by Stephen S.W. Lam                            **/
/** ------------------------------------------------------------------- **/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <dos.h>
#include "serial.h"

#define CR 13

void send_line(char *s);

void send_line(char *s)
{
    char *sp, c;

    for ( sp=s; *sp!='\0'; sp++)
    {
        c = *sp;
        writecomm(c);
        delay(20);
    }
    writecomm( CR );
    delay(200);          /** set delay period to allow PDP-11 to catch up **/
}

/* ****************************************
 * Serial Port Communications Functions  *
 ****************************************/

/* -----------initialize the com port ------------*/
void initcomport(void)
{
        BYTE abyte;

/*  Save the old UART registers */

        lctrl   = inportb(LINECTL);
        abyte   = (lctr.   DLAB);
        outportb(LINECTL,abyte);
        baudlsb = inportb(DIVLSB);
        baudmsb = inportb(DIVMSB);
        abyte  |= DLAB;
        outportb(LINECTL,abyte);
        inten   = inportb(INTENABLE);
        mctrl   = inportb(MODEMCTL);
        modstat = inportb(MODEMSTATUS);
        linstat = inportb(LINESTATUS);

/* Set up UART registers */
        abyte   = inportb(0x21);
        outportb(0x21,(abyte | 0x10));   /* disable interrupt handling, so
```

```
                                that interrupt vector address can be changed */
        abyte   = _intron();                /* redirects interrupt handler  */
        disable();                          /* disable interrupt processing */
        abyte   = inportb(0x21);                  /* enable Uart interrupts */
        outportb(0x21,(abyte & 0xef));

        outportb(LINECTL, 0x03);                                  /* set LINECTL */
        abyte   = inportb(RXDATA);          /* flush the UART recieve buffer  */
        outportb(INTENABLE,0x01);           /* select interrupts to catch     */
        outportb(MODEMCTL,0x0b);            /* set OUT2, RTS, DTR             */

/* SET BAUD RATE */
        abyte   = (inportb(LINECTL) | DLAB);                      /* set UART */
        outportb(LINECTL,abyte);
        outport(DIVLSB,0x18);                       /* set baud rate for 4800*/
        outport(DIVMSB,0x00);
        abyte  ^= DLAB;
        outportb(LINECTL,abyte);

        clear_serial_queue();
        enable();
        return;
}

/* --------restore the serial interrupt vector----------*/
void restore_serialint(void)
{
    BYTE abyte;
        abyte = (inportb(LINECTL) | DLAB);
        outportb(LINECTL,abyte);
        outport(DIVLSB,baudlsb);
        outport(DIVMSB,baudmsb);
        abyte ^= DLAB;
        outportb(LINECTL,abyte);

        outportb(LINECTL, lctrl);
        outportb(INTENABLE, inten);
        outportb(MODEMCTL, mctrl);

        _introff();
}

/* ----- clear the serial input buffer ---------- */
void clear_serial_queue(void)
{
    nextin = nextout = recvbuff;
    buffer_count = 0;
}

/* -------serial input interrupt service routines--------- */
int _intron(void)
{
        oldcomint = getvect(intp);
        setvect(intp, _irq4);
        return (0);
}

int _introff(void)
{
       if (oldcomint != NULL){
           setvect(intp, oldcomint);
           outportb(0x21,(inportb(0x21) | (0xff ^ 0xef)));
```

```
        }
        return (0);
}

void interrupt  _irq4()
{
        int CharIn;

        inportb(INTIDENT);                  /* clears Uart register */
        outportb(0x20,0x20);                        /* interrupt received */
        if (nextin == (recvbuff+BUFSIZE))
            nextin = recvbuff;                          /* circular buffer */
        CharIn = inportb(RXDATA);                        /* read the input */
        disable();
        if (xonxoff_enabled)
            if (CharIn == XOFF)                                 /* test XON */
                waiting_for_XON = 1;
            else if (CharIn == XON)                            /* test XOFF */
                waiting_for_XON = 0;
        if (!xonxoff_enabled || (CharIn != XON && CharIn != XOFF))   {
            *nextin++ = (char) CharIn;              /* put char in buffer   */
            buffer_count++;
           }
        if (xonxoff_enabled && !waiting_to_send_XON &&
            buffer_count > THRESHOLD)   {
           while ((inportb(LINESTATUS) & XMIT_DATA_READY) == 0)
              ;
           outportb(TXDATA, XOFF);                              /* send xoff */
           waiting_to_send_XON = 1;
          }
        enable();                           /* enable interrupt processing */
}

        /***********************************************
         *       Buffer character handling             *
         ***********************************************/

/* -----------read a character from the input buffer-------- */
int readcomm(void)
{
    if (nextout == recvbuff+BUFSIZE)
        nextout = recvbuff;
    --buffer_count;
    if (waiting_to_send_XON && buffer_count < SAFETYLEVEL) {
        waiting_to_send_XON = 0;
        writecomm(XON);
    }
    return *nextout++;
}

/* -------write a character to the comm port --------*/
int writecomm(int c)
{
int b;
    outportb(0x20,0x20);
    outportb(TXDATA, c);
    return TRUE;
}
```

APPENDIX E

THE PROGRAM HEADER FILE SERIAL.H

```
/** ----------------------- FILE: SERIAL.H ------------------------------ **/
/**  Header file required by JINCOMM.C                                    **/
/**  Modified from TCOMM.C by Stephen S.W. Lam                            **/
/** ---------------------------------------------------------------------- **/

/*                                      */
/*-------- serial port definitions -----*/
/*                                      */
extern int COMPORT;
extern char *nextin, *nextout;
/*-------------serial prototypes--------*/
void initcomport(void);
int readcomm(void);
int writecomm(int);
void clear_serial_queue(void);
void restore_serialint(void);
/* -------------macros------------------*/
#define comstat() (inp(LINESTATUS))
#define input_char_ready()  (nextin!=nextout)
#define XON     17
#define XOFF    19
/* ------------serial port addresses---------*/
/* -8250 uart base port address: COM1 3f8, com2 2f8 */
#define BASEPORT    (0x3f8)
#define TXDATA      BASEPORT        /* transmit data    */
#define RXDATA      BASEPORT        /* receive data     */
#define DIVLSB      BASEPORT        /* baud rate divisor lsb    */
#define DIVMSB      (BASEPORT+1)    /* baud rate divisor msb    */
#define INTENABLE   (BASEPORT+1)    /* interrupt enable */
#define INTIDENT    (BASEPORT+2)    /* interrupt ident  */
#define LINECTL     (BASEPORT+3)    /* line control     */
#define MODEMCTL    (BASEPORT+4)    /* modem control    */
#define LINESTATUS  (BASEPORT+5)    /* line status      */
#define MODEMSTATUS (BASEPORT+6)    /* modem status     */
/* -------------serial interrupt stuff ------------*/
#define IRQ     (4-(COMPORT-1))     /* 0-7 = IRQ0-IRQ7  */
#define COMINT  0x0c    /* interrupt vector 12/11   */
#define COMIRQ  (~(1<< IRQ))
#define PIC01   0x21    /* 8259 Programmable interrupt controller  */
#define PIC00   0x20    /* --------as above----------------------   */
#define EOI     0x20    /* End of interrupt command */
#define TIMER   0x1c    /* PC timer interrupt vector            */
/* -------------line status register values----*/
#define XMIT_DATA_READY 0x20
/* ---------interrupt enable register signals ---------*/
#define DATAREADY 0x09
/* --------serial input interrupt buffer---------------*/
#define BUFSIZE 1024
#define SAFETYLEVEL (BUFSIZE/4)
#define THRESHOLD (SAFETYLEVEL*3)
#define TRUE    1
#define FALSE   0

#define DLAB 0x80
 /* menu variables */
typedef unsigned char BYTE;
```

```
BYTE intem, mctrl, lctrl;          /* store existing UART settings */
unsigned int baudlsb, baudmsb;
volatile unsigned modstat;
volatile unsigned linstat;
/* functions prototypes */
void initcompert(void);
void restore_serialint(), clear_serial_queue();
int  _intron(void);
int  _introff(void);
void interrupt _irq4();
static void(interrupt *oldcomint)(void);

int readcomm(void);
int writecomm(int);
/* variables */
extern int COMPORT;

char recvbuff[BUFSIZE];
char *nextin = recvbuff;
char *nextout = recvbuff;
int  buffer_count;

int   COMPORT = 1;    /* COM1 or 2                */
int   PARITY  = 0;    /* 0=none, 1=odd, 2=even    */
int   STOPBITS = 1;   /* 1 or 2                   */
int   WORDLEN = 8;    /* 7 or 8                   */
int   BAUD    = 4800; /* 1200 2400 4800 9600      */

int         xonxoff_enabled = TRUE;
static int  waiting_for_XON;
static int  waiting_to_send_XON;

int intp = 0x0c;           /* Interrupt vector #12 for COM 1 */
```

APPENDIX F

THE PROGRAM SOURCE FILE JINDIVIK.FTN

```
      PROGRAM JINDIV

C*****************************************************************************
C
C  This program communicates with JIN1 and JIN2 on the IBM PS/2-80
C  for the JINDIVIK Auxilary Intake pressure measurements.
C
C  The program is lauched by JIN1 from the PS/2.  It asks the operator
C  at the Venturi gauge to key in the parameters.
C  It then signals the PS/2 to start scanning the pressure values.
C  The pressure values are multiplied by 1000 so as to be passed
C  back onto the PDP-11 in integer form and stored in .JCS and .DAT
C  files and are compatible with the previous tests.
C
C  Written by:
C
C       Stephen S.W. Lam
C       Applied Aerodynamics Group
C       Flight Mechanics & Propulsion Division
C
C       22-OCT-1991
C
C*****************************************************************************

      CHARACTER key*1
      CHARACTER title*25, custom*20, tpn*3, surnam(3)*8, desc*80,
     1          confnm*20, filnam*20, fildat*11, jcsfil*20
      REAL    pa, ta, p1, t1, pd
      INTEGER ipress(59), len, iser
      BYTE    bell
      PARAMETER (iser=470)    ! serial no. of test, any number will do

      DATA    title/'JINDIVIK AUXILIARY INTAKE'/
      DATA    custom/'PROPULSION BRANCH'/
      DATA    bell/7/
      DATA    confnm/'CONFJIN.DAT'/

C-----------------------------------
C  Define Test Point Number to be 1
C         Config number = 1
C-----------------------------------

      tpn(1:) = '1 '
      konfig = 1

      filnam(1:4)='DL1:'
C-----------------------------------------------
C  Open terminal line TT6: for communication with
C  the operator at the Venturi
C-----------------------------------------------
      OPEN (unit=1, file='tt6:', status='old')

C-----------------------------------------
C  Receive test information from the PS/2
C-----------------------------------------
      READ ( 5, 10 ) len, desc(1:len)
```

```
      READ ( 5, 10 ) namlen, filnam(5:namlen+4)
 10   FORMAT( q, A )
      READ ( 5, 20 ) fildat
 20   FORMAT ( A )
      namlen=namlen+4
      jcsfil = filnam
      jcsfil(namlen+1:) = '.JCS'
      filnam(namlen+1:) = '.DAT'

C---------------------------------------
C  Ask the operator to key in parameters
C---------------------------------------
      WRITE ( 1, 25) bell, desc(1:lem)
 25   FORMAT(//, x, 72('-') /, x, a1, /, x, a, //,
     1       ' Ready to Take Data: ...' )

 30   CONTINUE
      WRITE ( 1, 50 )
 50   FORMAT ('   P1 (in. of water) = ', $)
      READ ( 1, * ) p1
      WRITE ( 1, 60 )
 60   FORMAT ('   T1 (deg C) = ', $)
      READ ( 1, * ) t1
      WRITE ( 1, 65 )
 65   FORMAT ('   Pressure Drop (in. of water) = ', $)
      READ ( 1, * ) pd

      WRITE ( 1, 70 )
 70   FORMAT ( /,' Are the above values entered correctly [Y/N]? ',$)
      READ ( 1, 20 ) key
      IF ( (key .eq. 'N') .OR. (key .eq. 'n') ) GO TO 30
      WRITE( 1, 75 )
 75   FORMAT( /' Sampling data, Please wait ...',$ )

C ------------------------------
C  Send parameters back to PS/2
C ------------------------------
      WRITE ( 5, 80 ) t1, p1, pd
 80   FORMAT( 3(x,f6.2), ' C' )    ! 'C' signals end of data line
      READ ( 5, 20 ) key
      IF (key .eq. 'C') then
          READ ( 5, * ) alpha, beta, pa, ta

C-------------------------------------------------
C  Read pressure data in 3 lines of 20 values each
C-------------------------------------------------
          READ ( 5, * ) ( ipress (i), i = 1,20 )
          READ ( 5, * ) ( ipress (i), i = 21,40 )
          READ ( 5, * ) ( ipress (i), i = 41,59 )

C-----------------------------------------------------------
C  Confirm to console that all pressure data have been read
C-----------------------------------------------------------
          WRITE ( 5, 90 )
 90       FORMAT ( x, 1H> )
C-------------------------------------------------------------------------
C  Calculate the dynamic pressure, q
C
C    ipress(7) and ipress(8) both measures the tunnel total pressure
C    ipress(5) and ipress(6) both measures the tunnel static pressure
C    Results of HmP ( H - P ) is divided by 1000 to convert to kPa
C    then multiplied by 0.5 to take the average.
```

```
C-----------------------------------------------------------------------
            E = Pa+0.0005*(ipress(7)+ipress(8))*6.895
            Ps = Pa+0.0005*(ipress(5)+ipress(6))*6.895
            EmP = E-Ps
C           EmP = 6.895*EmP      ! convert from psi to kPa
            q = 3.5*Ps*((EmP/Ps+1.0)**(2/7)-1.0)

C -------------------------
C  Write data to .JCS file
C -------------------------
            OPEN (unit=2, file=jcsfil, status = 'new')
            WRITE ( 2, * ) (ipress (i), i=1,59 )
            WRITE ( 2, * ) alpha, beta, ti, pi, pd, pa, ta
            WRITE ( 2, 100 ) desc(1:lem)
 100        FORMAT ( /, x, A<lem> , ///)
            WRITE ( 2, 110 ) title
            WRITE ( 2, 110 ) custom
            WRITE ( 2, 110 ) tpa
            WRITE ( 2, 110 ) filnam
            WRITE ( 2, 110 ) fildat
            WRITE ( 2, 110 ) confnm
            WRITE ( 2, 120 ) konfig
 110        FORMAT (A)
 120        FORMAT ( i4 )
            CLOSE (2)

C -------------------------
C  Write data to .DAT file
C -------------------------
            OPEN (unit=3, file=filnam, status = 'new')
            WRITE( 3, 150 ) filnam, fildat
 150        FORMAT( x, A, /, x, A )
            WRITE( 3, 160 ) int(pa*10), iser, q*1000
 160        FORMAT( x, I4, x, I3, ' ?? ???? ???? ', F6.1 /,
       1        x, '111111111 0000000  000000  00000000  00000000', /,
       2        x, '000000 000000  A:: B:: C:: D:: E:: F::')
            WRITE( 3, 170 ) konfig, alpha, beta, ti, pi, dp, pa, ta
 170        FORMAT(x, '00 ',i2.2, 5(x,spf6.2), 2(x,spf7.2) )
            WRITE( 3, 180 ) ( ipress(i), i = 1, 59 )
 180        FORMAT( 8 ( 2x, spi5.4 ) )
       ELSE
            WRITE ( 1, 130) bell
 130        FORMAT (//, x, a1, 'Run Aborted'/ )
       END IF

       WRITE ( 1, 140 ) bell
 140   FORMAT(//, x, a1, 'End of Run '/)
       END
```

APPENDIX G

AIR MASS FLOW RATE EQUATION

The working equation for the engine air mass flow rate is derived from Section 1.1 of the British Standard 1042 (1981) [1]. The basic equations for the mass flow rate of a fluid through a venturi tube is given by

$$\dot{m} = CE\epsilon\frac{\pi}{4}d^2\sqrt{2\Delta P \times \rho_1} \qquad (1)$$

where C is the discharge coefficient,
E is the velocity of approach factor, $E = (1-\tau^4)^{-\frac{1}{2}}$,
τ is the diameter ratio, $\tau = \dfrac{d}{D}$,
d is the diameter of the venturi throat,
D is the upstream internal pipe diameter of the venturi tube,
ϵ is the expansibility factor, $\epsilon = \left[\left(\dfrac{\gamma\lambda^{\frac{2}{\gamma}}}{\gamma-1}\right)\left(\dfrac{1-\tau^4}{1-\tau^4\lambda^{\frac{2}{\gamma}}}\right)\left(\dfrac{1-\lambda^{\frac{\gamma-1}{\gamma}}}{1-\lambda}\right)\right]^{\frac{1}{2}}$,
γ is the ratio of specific heat capacities,
λ is the pressure ratio, $\lambda = 1 - \dfrac{\Delta P}{P_1}$,
P_1 is the pressure upstream of the venturi throat,
ΔP is the pressure drop across the venturi throat, and
ρ_1 is the fluid density upstream of the venturi throat.

The throat diameter d of the venturi tube is 142.75 mm and the upstream internal pipe diameter D is 304.30 mm. The tube has a rough-cast convergent section to which the following parameter values are applied:—

$$C = 0.984\ , \quad \tau = 0.469\ , \quad E = 1.025\ , \quad \gamma = 1.4\ .$$

With these values equation (1) may be simplified to:—

$$\dot{m} = 0.0228\epsilon\sqrt{\Delta P \times \rho_1}\ . \qquad (2)$$

It should be noted that the value of the venturi upstream pressure, say P', as recorded by the operator is in inches of H_2O and is relative to the tunnel total pressure P_o. P_1 is then calculated from P' according to the following equation:—

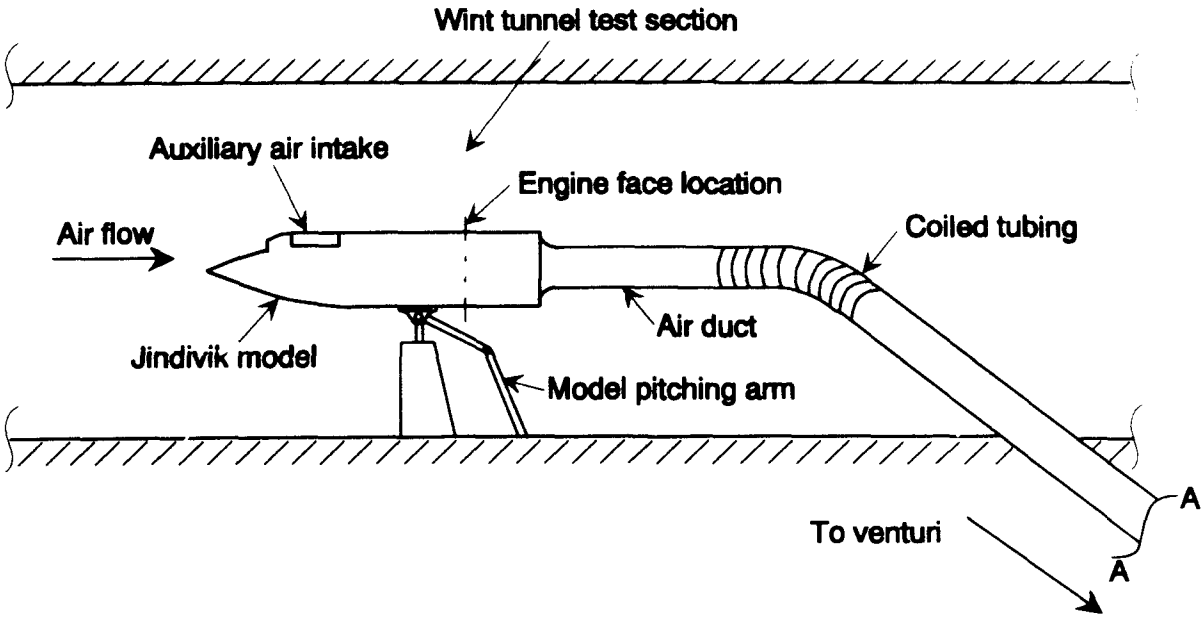
$$P_1 = P_o - 0.249P'\ .$$

The density ρ_1 of air upstream of the venturi throat is calculated from the measured temperature T_1 and P_1 using the real gas equation:—

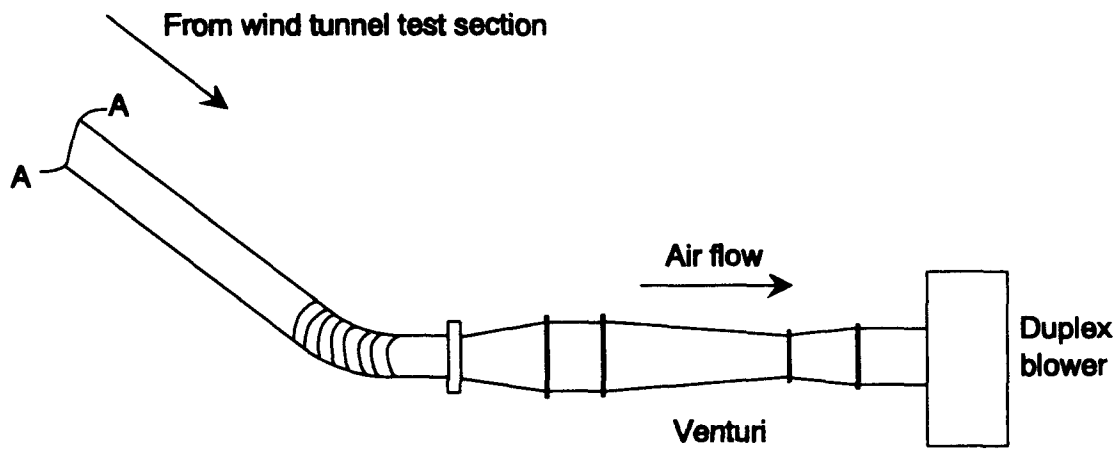
$$\rho_1 = \frac{P_1}{RT_1}\ ,$$

where R is the gas constant of air and is equal to 287.04 J $kg^{-1}K^{-1}$.

[1] **British Standard 1042: Section 1.1: 1981.** ***Methods of measurement of fluid flow in closed conduits. Part 1. Pressure differential devices. Section 1.1 Orifice plates, nozzles, and venturi tubes inserted in circular cross-section conduits running full.*** **British Standards Institution, UK.**



(a) Jindivik model with engine air duct attachment in the wind tunnel test section.



(b) The venturi and duplex blower assembly.

Figure 1: Schematic of the experimental set-up of the Jindivik air intake model in the wind tunnel.

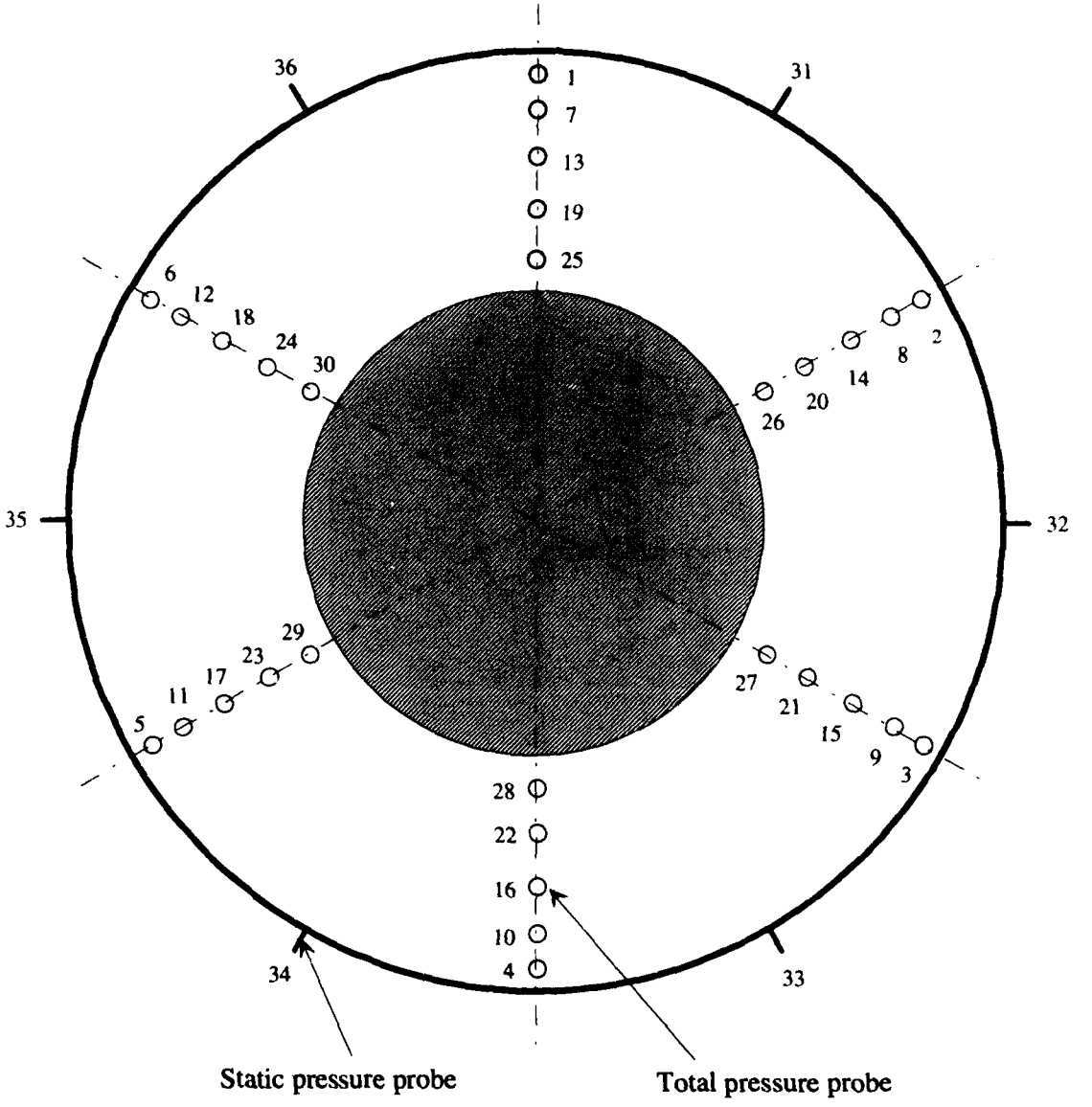
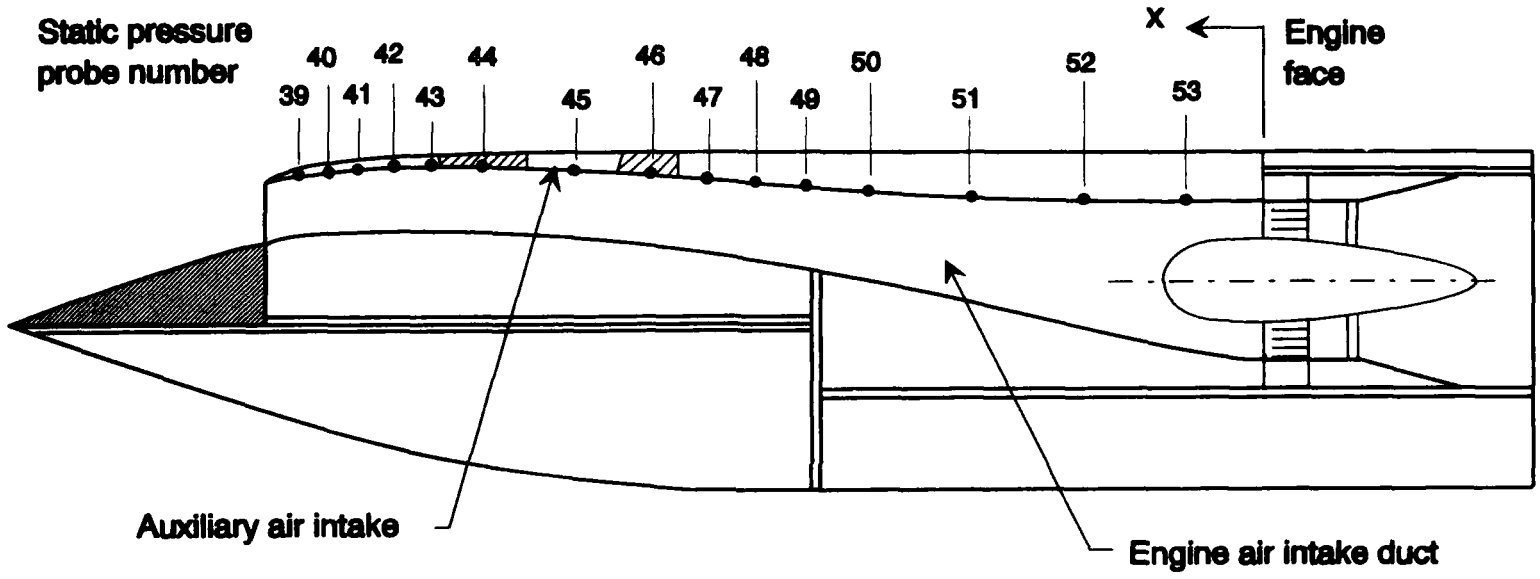


Figure 2: Positions and numbering scheme of the pressure probes at the engine face of the Jindivik air intake duct.



Probe	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53
x(m)	0.680	0.660	0.640	0.610	0.580	0.490	0.435	0.395	0.380	0.365	0.335	0.275	0.215	0.155	0.095

Figure 3: Locations of the static pressure probe along the air intake duct of a Jindivik forward fuselage model.

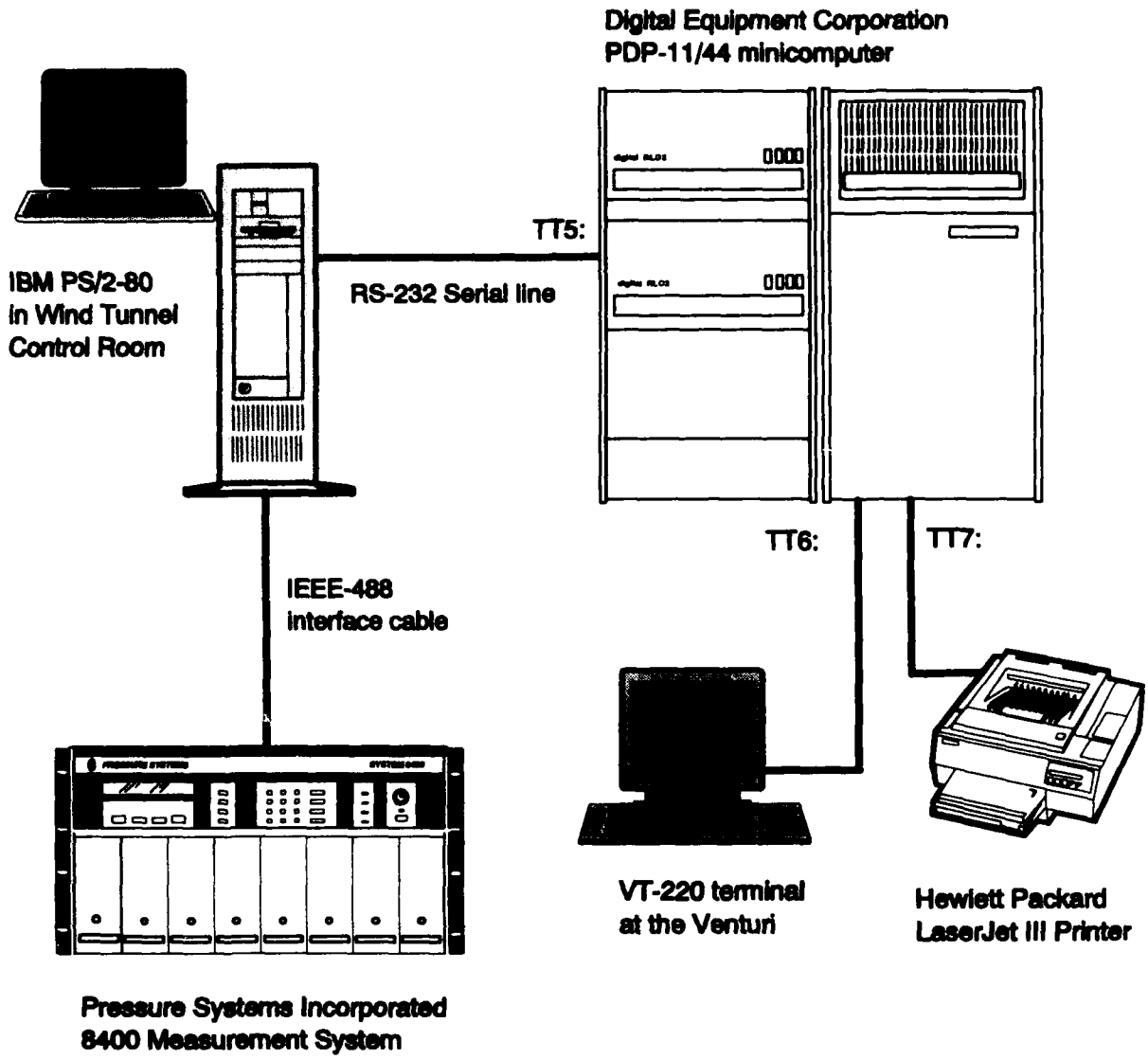


Figure 4: Schematic of the layout of the data acquisition system for the Jindivik auxiliary intake test in the Low Speed Wind Tunnel.

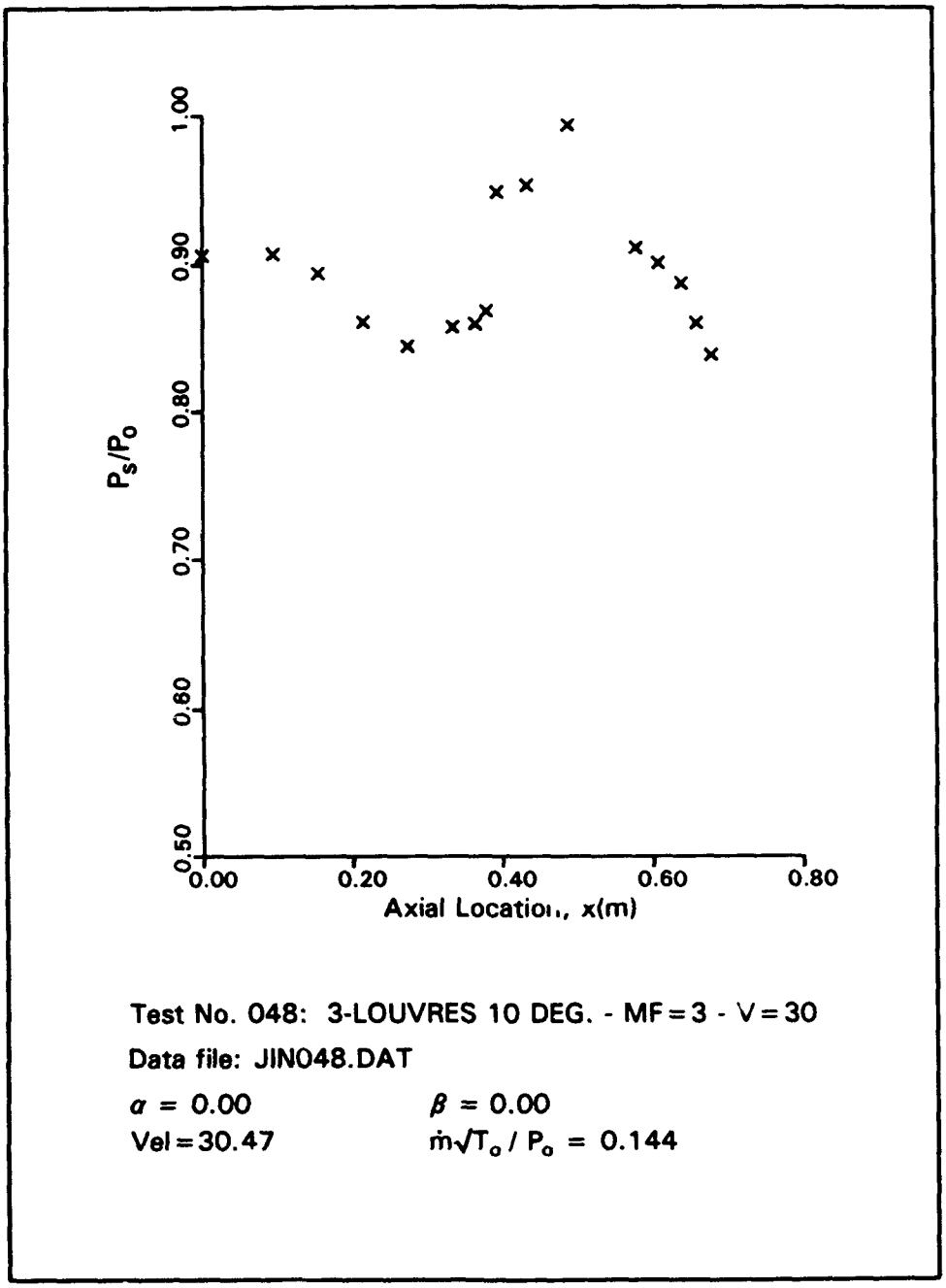


Figure 5: An example of the graphical output of the static to total pressure ratio versus the axial location along the intake duct away from the engine face towards the intake lip.

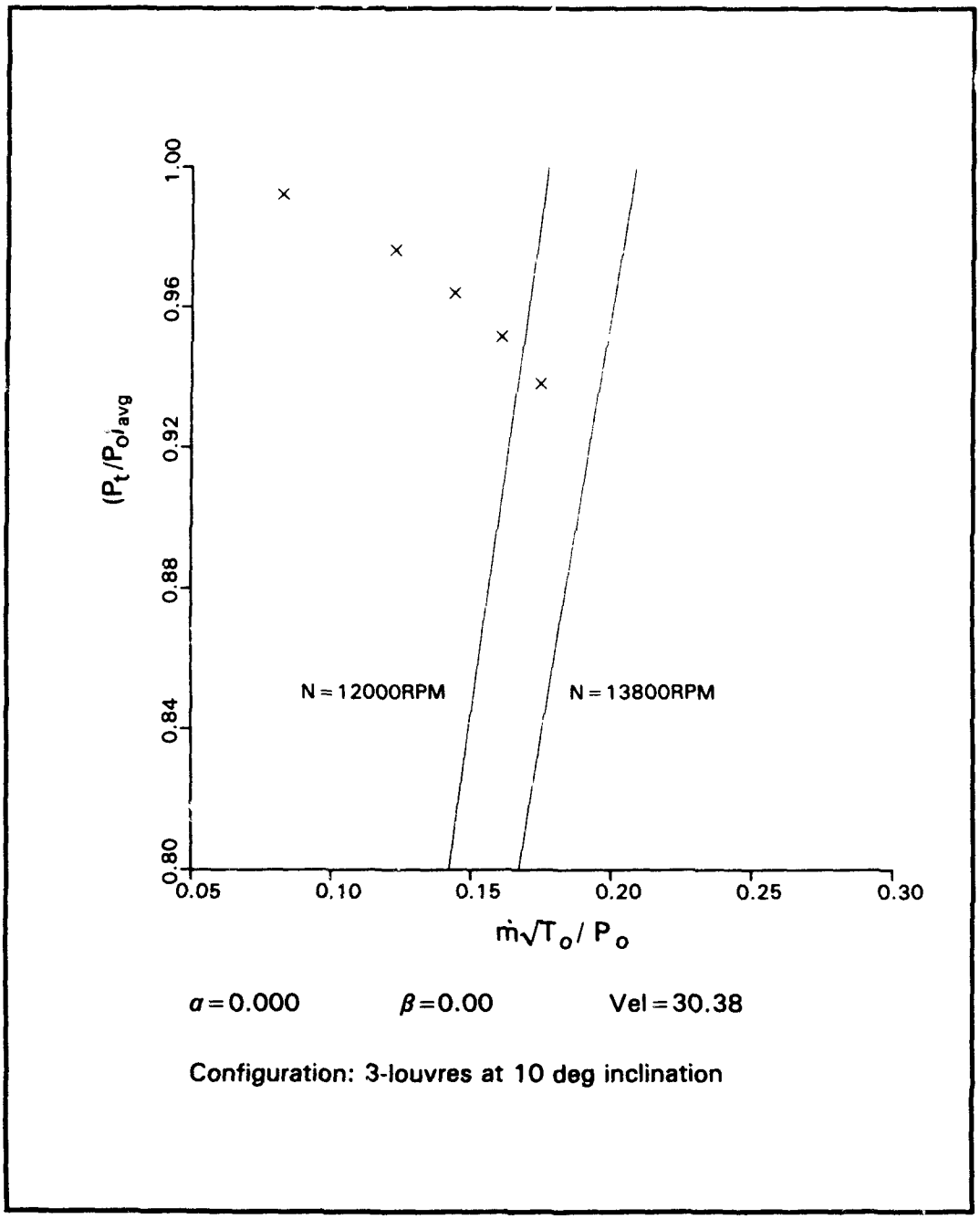


Figure 6: An example of the graphical output of the pressure recovery versus engine mass flow parameter for a given auxiliary air intake configuration.

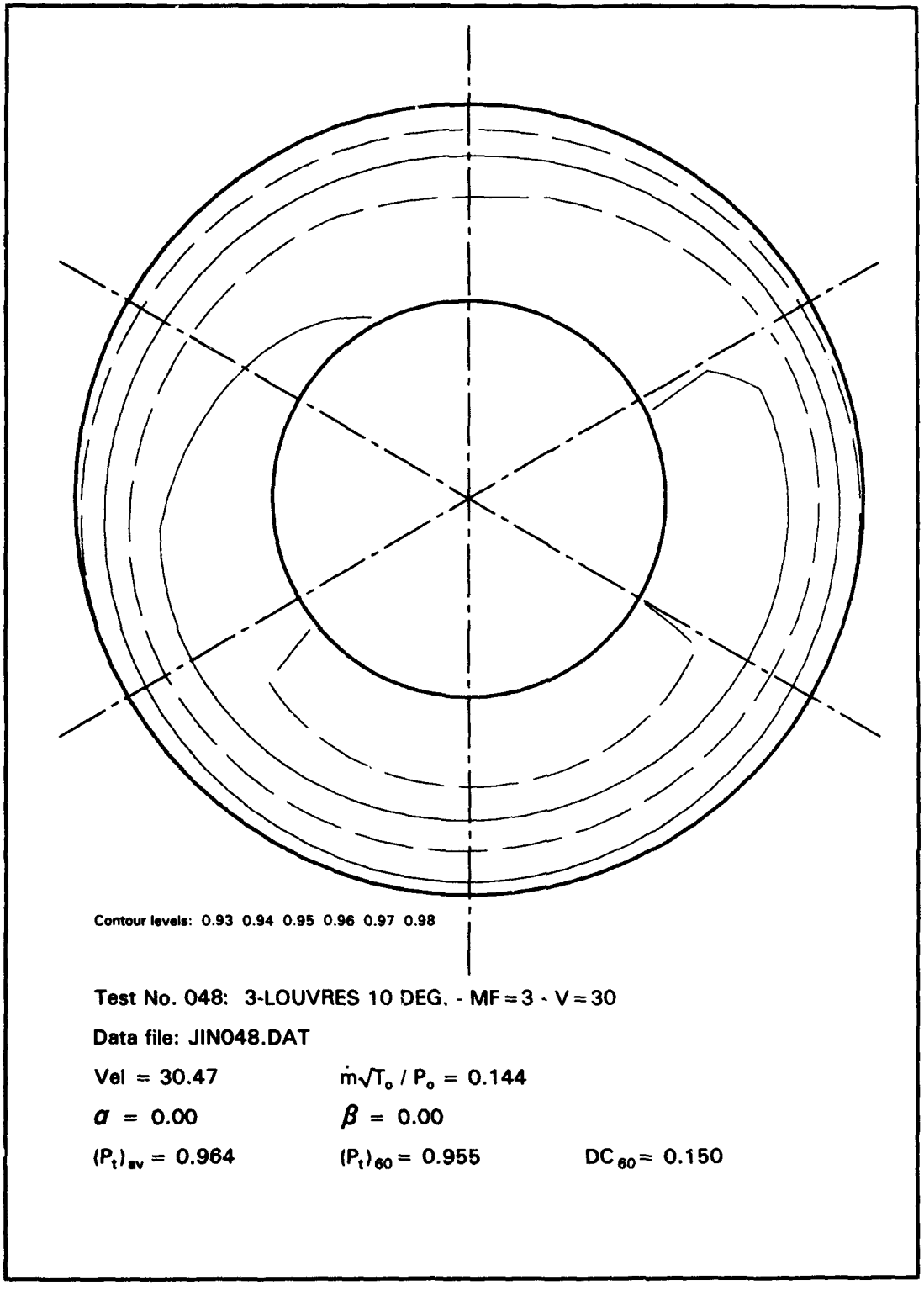


Figure 7: An example of the sectional contour plot of the total pressure at the engine face annulus. The values of the contour level are increasing from the inner surface outward.

AERONAUTICAL RESEARCH LABORATORY
LOW SPEED WIND TUNNEL

Test Results on
JINDIVIK AUXILIARY INTAKE

Test Number: 48
3-LOUVRES 10 DEG. - MF=3 - V=30

Data File: DL1:JIN048.DAT Created on 15-NOV-1991

VELOCITY = 30.47
ALPHA = 0.00
BETA = 0.00

ANNULUS NO. : 1

PROBE	PT/P0	PV/P0
1	0.9360	0.0297
2	0.9428	0.0365
3	0.9480	0.0417
4	0.9508	0.0445
5	0.9477	0.0414
6	0.9399	0.0336
AVG	0.9442	0.0379

ANNULUS NO. : 2

PROBE	PT/P0	PV/P0
7	0.9488	0.0425
8	0.9595	0.0532
9	0.9610	0.0547
10	0.9612	0.0549
11	0.9618	0.0555
12	0.9523	0.0460
AVG	0.9574	0.0511

ANNULUS NO. : 3

PROBE	PT/P0	PV/P0
13	0.9590	0.0527
14	0.9684	0.0621
15	0.9739	0.0676
16	0.9730	0.0667
17	0.9731	0.0668
18	0.9638	0.0575
AVG	0.9685	0.0622

ANNULUS NO. : 4

PROBE	PT/P0	PV/P0
19	0.9630	0.0567
20	0.9697	0.0634
21	0.9796	0.0733
22	0.9846	0.0783
23	0.9781	0.0718
24	0.9688	0.0625
AVG	0.9740	0.0677

ANNULUS NO. : 5

PROBE	PT/P0	PV/P0
25	0.9657	0.0594
26	0.9695	0.0632
27	0.9798	0.0735
28	0.9864	0.0801
29	0.9787	0.0724
30	0.9739	0.0676
AVG	0.9757	0.0694

Pstatic - ENGINE FACE

PROBE	PS/P0
31	0.9033
32	0.9049
33	0.9087
34	0.9064
35	0.9093
36	0.9053

AX POS	1	2	3	4	5	6	7	8
PS/P0	0.906	0.907	0.894	0.861	0.845	0.858	0.860	0.868

AX POS	9	10	11	12	13	14	15	16
PS/P0	0.550	0.550	0.550	0.911	0.901	0.887	0.860	0.838

PT60 = 0.95528 DC60 = 0.15045 PT60PS = 66 PTAV = 0.96396
PVAV = 0.05766 PSAV = 0.90630 RO = 0.06910 EPS = 0.99198
MDOT = 0.86007 B = 1007.700 TA = 288.25 DP = 5.25
P1 = 29.70 T1 = 20.00 Ptu = 101.277 XMDOT = 0.14418

Figure 8: An example of a printed output of a test summary produced by CALJIN.

Data Acquisition and Processing Software for the Low Speed Wind Tunnel Tests of the Jindivik Auxiliary Air Intake

S.S.W. Lam

DSTO-TR-0043

DISTRIBUTION

AUSTRALIA

DEFENCE ORGANISATION

Defence Science and Technology Organisation

Chief Defence Scientist
FAS Science Policy
AS Science Corporate Management
} shared copy

Counsellor Defence Science, London (Doc Data Sheet only)
Counsellor Defence Science, Washington (Doc Data Sheet only)
Senior Defence Scientific Adviser (Doc Data Sheet only)
Scientific Advisor Policy and Command (Doc Data Sheet only)
Navy Scientific Adviser (3 copies Doc Data Sheet only)
Scientific Adviser - Army (Doc Data Sheet only)
Air Force Scientific Adviser (Doc Data Sheet only)
Scientific Adviser to Thailand MRD (Doc Data sheet only)
Scientific Adviser to the DRC (Kuala Lumpur) (Doc Data sheet only)

Aeronautical and Maritime Research Laboratory
Director
Library Fishermens Bend
Library Maribyrnong
Chief Air Operations Division
Author: S.S.W. Lam
N. Matheson (3 copies)
M. Glaister
L. MacLaren
Y.Y. Link
P. Malone
A. Abdel-Fattah

Electronics and Surveillance Research Laboratory
Director
Main Library - DSTO Salisbury

Defence Central

OIC TRS, Defence Central Library
Document Exchange Centre, DSTIC (8 copies)
Defence Intelligence Organisation
Library, Defence Signals Directorate (Doc Data Sheet Only)
FAS Industry Policy and Operations

HQ ADF

Director General Force Development (Air)

Navy

Aircraft Maintenance and Flight Trials Unit
Director Aircraft Engineering - Navy
Director of Aviation Projects - Navy
Director of Naval Architecture
RAN Tactical School, Library
Superintendent, Naval Aircraft Logistics

Army

US Army Research, Development and Standardisation Group (3 copies)

Air Force

Aircraft Research and Development Unit
Tech Reports,CO Engineering Squadron,ARDU
OIC ATF, ATS, RAAFSTT, WAGGA (2 copies)

UNIVERSITIES AND COLLEGES

Australian Defence Force Academy
Library
Head of Aerospace and Mechanical Engineering

Adelaide
Barr Smith Library

Flinders
Library

LaTrobe
Library

Melbourne
Engineering Library

Monash
Hargrave Library

Newcastle
Library

New England
Library

Sydney
Engineering Library

NSW
Physical Sciences Library

Queensland
Library

Tasmania
Engineering Library

Western Australia
Library

RMIT
Library

University College of the Northern Territory
Library

OTHER GOVERNMENT DEPARTMENTS AND AGENCIES

AGPS

OTHER ORGANISATIONS

NASA (Canberra)
ASTA Engineering, Document Control Office
Hawker de Havilland Aust Pty Ltd, Victoria, Library

SPARES (10 COPIES)

TOTAL (70 COPIES)

AL 149 DEPARTMENT OF DEFENCE

PAGE CLASSIFICATION
UNCLASSIFIED

PRIVACY MARKING
—

DOCUMENT CONTROL DATA

1a. AR NUMBER	1b. ESTABLISHMENT NUMBER	2. DOCUMENT DATE	3. TASK NUMBER
AR-007-100	DSTO-TR-0043	AUGUST 1994	DST 92/459

4. TITLE

DATA ACQUISITION AND PROCESSING SOFTWARE FOR THE LOW SPEED WIND TUNNEL TESTS OF THE JINDIVIK AUXILIARY AIR INTAKE

5. SECURITY CLASSIFICATION (PLACE APPROPRIATE CLASSIFICATION IN BOX(S) IE. SECRET (S), CONF. (C) RESTRICTED (R), LIMITED (L). UNCLASSIFIED (U)).

DOCUMENT	TITLE	ABSTRACT
U	U	U

6. NO. PAGES
43

7. NO. REFS.
5

8. AUTHOR(S)

S.S.W. LAM

9. DOWNGRADING/DELIMITING INSTRUCTIONS

Not applicable.

10. CORPORATE AUTHOR AND ADDRESS

AERONAUTICAL AND MARITIME RESEARCH LABORATORY
AIR OPERATIONS DIVISION
GPO BOX 4331
MELBOURNE VIC 3001 AUSTRALIA

11. OFFICE/POSITION RESPONSIBLE FOR:

DSTO

SPONSOR _____

SECURITY _____

DOWNGRADING _____

CAOD

APPROVAL _____

12. SECONDARY DISTRIBUTION (OF THIS DOCUMENT)

Approved for public release.

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DSTIC, ADMINISTRATIVE SERVICES BRANCH, DEPARTMENT OF DEFENCE, ANZAC PARK WEST OFFICES, ACT 2601

13a. THIS DOCUMENT MAY BE ANNOUNCED IN CATALOGUES AND AWARENESS SERVICES AVAILABLE TO ....

No limitations.

14. DESCRIPTORS

Air intakes
Jindivik aircraft
AMRL low speed wind tunnel
Wind tunnel tests
Data acquisition
Computer programs

15. DISCAT SUBJECT CATEGORIES
010310
010101

16. *ABSTRACT*

Data acquisition software has been developed for the wind tunnel tests of the auxiliary air intake of the Jindivik pilotless target aircraft in the AMRL Low Speed Wind Tunnel (LSWT) using the new Pressure Systems Incorporated (PSI) 8400 Measurement System under the control of an IBM PS/2 computer. The recent upgrade of the data acquisition system for the LSWT and the replacement of the mechanical Scanivalve pressure measuring system with PSI electronic pressure scanners has required major modifications to be made to existing software. To minimise the changes needed and to provide compatibility with data processing for previous tests the data acquired by the PSI electronic pressure equipment are transferred to the dedicated LSWT DEC PDP-11/44 mini-computer for storage and processing. This report describes the development and operation of new software for the LSWT tests of the Jindivik auxiliary air intake.

PAGE CLASSIFICATION
UNCLASSIFIED

PRIVACY MARKING

THIS PAGE IS TO BE USED TO RECORD INFORMATION WHICH IS REQUIRED BY THE ESTABLISHMENT FOR ITS OWN USE BUT WHICH WILL NOT BE ADDED TO THE DISTIS DATA UNLESS SPECIFICALLY REQUESTED.

16. ABSTRACT (CONT).

17. IMPRINT

AERONAUTICAL AND MARITIME RESEARCH LABORATORY, MELBOURNE

18. DOCUMENT SERIES AND NUMBER	19. WA NUMBER	20. TYPE OF REPORT AND PERIOD COVERED
DSTO Technical Report 0043	54 527G	

21. COMPUTER PROGRAMS USED

22. ESTABLISHMENT FILE REF.(S)

M1/8/796

23. ADDITIONAL INFORMATION (AS REQUIRED)